**DEVELOPMENT PHASE I REQUIREMENTS REPORT**

For

COURSE REGISTRATION SYSTEM

Version 1.0

Prepared by: Team Innovators ( Sreekanth Vobilishetty, Akhila Yarlagadda, Piyusha Varshini Tirukovalluru, Meghana Reddy Akkati)

University of North Texas

10/22/2019

**1 COURSE ENROLLMENT REQUIREMENTS**

Course Enrollment should basically provide the opportunity to enroll for a class which the student wants to take. Here the student can enroll the courses of their choice by registering and logging into the website. his feature aims to develop the user authentication requirements and some of the functionalities like add, drop, swap, search for a course.

* 1. **User Registration**

Student first needs to register for this student portal to enroll for the courses of their choice, here the student should use their id’s given by the university to get registered into the portal. Once registered the student can use their id and password to get access to the student portal.

* 1. **User Login**

Everytime the student needs to login they should use the id and password they mentioned during the time of the registration. This login feature takes you to the main page where you can perform various functionalities that are necessary to get enrolled in a course.

* 1. **Add Feature**

Student should be able to add desired courses (which are available) to the shopping cart. But this add feature restricts you from adding the desired courses if there are any prerequisites for the respective student and if the desired course exceeds the limited strength of the class.

* 1. **Drop Feature**

Student should be able to drop any of the courses which he is already enrolled in before certain deadline.

* 1. **Swap Feature**

Student should also be allowed to swap an already enrolled course with some other course (available course) of his choice. The student can only swap with the courses that satisfies the conditions which are mentioned in the add feature.

* 1. **Enroll Feature**

After adding to the shopping cart, this is a final step to get registered. The student should be enrolled when he clicks ‘enroll’. If the class is full, the student must get added to the waiting list.

If the class capacity gets increased or if any other student who already enrolled drops from the course then student automatically gets enrolled into the course.

* 1. **Search Feature**

Student can search for any desired course by entering the course name or the course id. This feature makes it flexible for the student to search for a specific course instead of searching the entire course list, this feature filters the list of the course list.

* 1. **Course Catalogue**

Course catalogue displays all the courses available and also it shows the description of each course. It gives the student some basic idea regarding the course.

* 1. **Updated Requirements**

In software requirements *( Section 2.4.1)*  we have changed from JSP to angular 6, because there are some issues while rendering JSP with spring boot. Angular 6 is the advanced technology and makes the development more easier, directives feature in angular 6 allowed developers to assign special behaviors to create dynamic and rich content with HTML.

We also added the new requirement email, here an email is sent to students mail address automatically when the student drops/add/swap the course for the confirmation.

**Development phase I:**

In this phase we develop the basic user authentication feature for both the student and admin, because security is primary concern for any website, here the student can access the website with there valid credentials. We also decided to implement the complete enrollment features of the student. The functional requirements which are implemented in this are:

User Registration

User Login

Features to add the course

Features to drop the course

Features to swap the course

Features to search for a course

Features to finish enrolling a course

Features to view the course catalogue

**Development phase II:**

In this phase we enhance the features of the student, here we intended to include the finance operations performed by the student to get enrolled into the courses. We also intended to include the complete features to view the course details. The functional requirements to be implemented in this phase are:

Features to make payment for a course

Features to ow the account balance

Features to show the complete summary of the payment

Transaction history

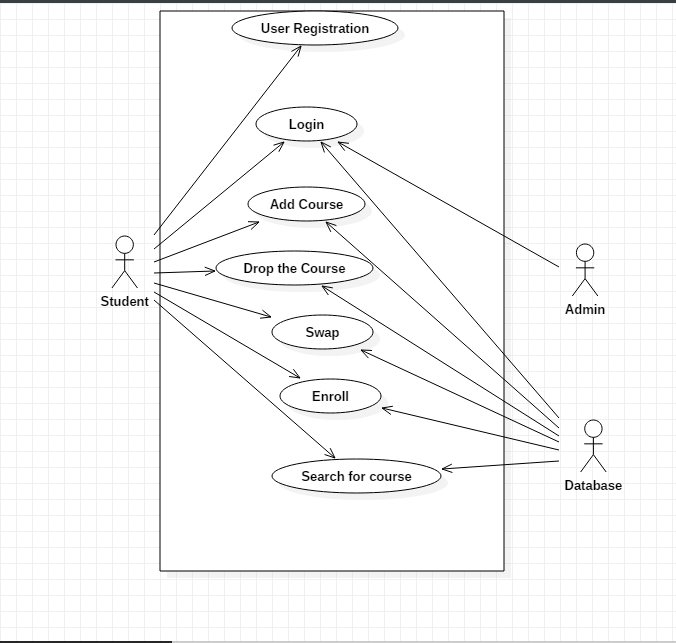
Features to view the class schedule

Features to view the grades to student

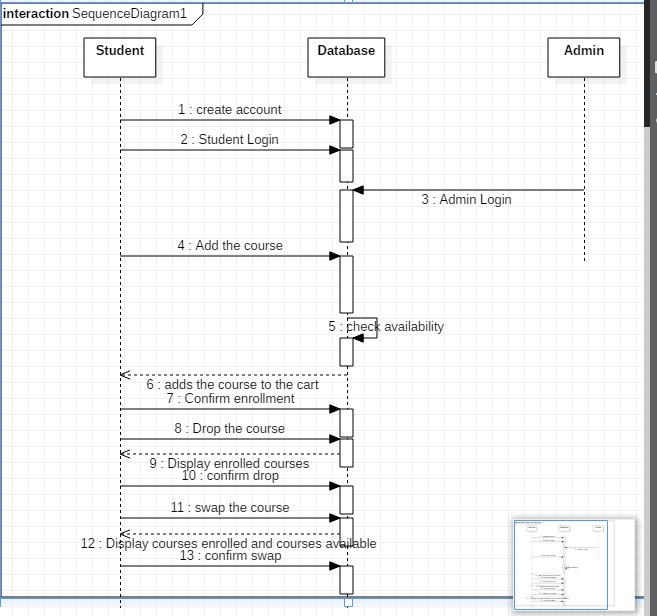
Features to view the mandatory courses

**2. UML DIAGRAMS**

**2.1 Usecase Diaram**

****

**2.2 Sequence Diagram**

****

**3.TESTCASES**

4.UNIT TESTING WITH SPRING BOOT AND JUNIT

4.1 Unit Testing

We want to create a Unit test for the Controller class which is Registration Controller in our project which has several Get and Post methods which needs to be tested.

In the Unit testing,

We will mock out the Registration Service using Mockito.

We will use Mock MVC framework to launch only the Registration Controller.

The following methods used in the Controller are:



4.2 JUnit Test code

package com.unt.registration.controller;

//package com.javainuse.test;

import static org.junit.Assert.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import org.junit.runner.RunWith;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

import org.junit.Test;

import org.mockito.Mockito;

import org.skyscreamer.jsonassert.JSONAssert;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.http.MediaType;

import org.springframework.test.context.junit4.SpringRunner;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.MvcResult;

import org.springframework.test.web.servlet.RequestBuilder;

import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

import com.unt.registration.service.\*;

@RestController

public class RegistrationController {

@RunWith(SpringRunner.class)

@WebFluxTest(controllers = RegistrationController.class)

public class RegistrationControllerTest {

@Autowired

private RegistrationService registrationService;

@GetMapping("/students/{studentId}/courses")

public List<Course> getCourses(@PathVariable String studentId) {

return registrationService.fetchEnrolledCourses(user);

}

@GetMapping("/students/{studentId}/courses/{courseId}")

public Course retrieveDetailsForCourse(@PathVariable String studentId,

@PathVariable String courseId) {

return registrationService.fetchEnrolledCourses(user);

}

@Autowired

private MockMvc mockMvc;

@MockBean

private registrationService studentService;

Course mockCourse = new Course("Course1", "Spring", "10 Steps",

Arrays.asList("Learn Maven", "Import Project", "First Example",

"Second Example"));

String exampleCourseJson = "{\"name\":\"Spring\",\"description\":\"10 Steps\",\"steps\":[\"Learn Maven\",\"Import Project\",\"First Example\",\"Second Example\"]}";

@Test

public void retrieveDetailsForCourse() throws Exception {

Mockito.when(

studentService.retrieveCourse(Mockito.anyString(),

Mockito.anyString())).thenReturn(mockCourse);

RequestBuilder requestBuilder = MockMvcRequestBuilders.get(

"/students/Student1/courses/Course1").accept(

MediaType.APPLICATION\_JSON);

MvcResult result = mockMvc.perform(requestBuilder).andReturn();

System.out.println(result.getResponse());

String expected = "{id:Course1,name:Spring,description:10 Steps}";

// {"id":"Course1","name":"Spring","description":"10 Steps, 25 Examples and 10K Students","steps":["Learn Maven","Import Project","First Example","Second Example"]}

JSONAssert.assertEquals(expected, result.getResponse()

.getContentAsString(), false);

}

}

@Test

public void createStudentCourse() throws Exception {

Course mockCourse = new Course("1", "Smallest Number", "1",

Arrays.asList("1", "2", "3", "4"));

// studentService.addCourse to respond back with mockCourse

Mockito.when(

studentService.addCourse(Mockito.anyString(),

Mockito.any(Course.class))).thenReturn(mockCourse);

// Send course as body to /students/Student1/courses

RequestBuilder requestBuilder = MockMvcRequestBuilders

.post("/students/Student1/courses")

.accept(MediaType.APPLICATION\_JSON).content(exampleCourseJson)

.contentType(MediaType.APPLICATION\_JSON);

MvcResult result = mockMvc.perform(requestBuilder).andReturn();

MockHttpServletResponse response = result.getResponse();

assertEquals(HttpStatus.CREATED.value(), response.getStatus());

assertEquals("http://localhost/students/Student1/courses/1",

response.getHeader(HttpHeaders.LOCATION));

@Test

public void testUserValidate() {

}

@Test

public void testSignup() {

fail("Not yet implemented");

}

@Test

public void testFetchAllDepartments() {

fail("Not yet implemented");

}

@Test

public void testResetPassword() {

fail("Not yet implemented");

}

@Test

public void testGetCourses() {

fail("Not yet implemented");

}

@Test

public void testFindCourse() {

fail("Not yet implemented");

}

@Test

public void testEnroll() {

fail("Not yet implemented");

}

@Test

public void testDrop() {

fail("Not yet implemented");

}

@Test

public void testFetchEnrolledCourses() {

}

@Test

public void testSwap() {

fail("Not yet implemented");

}

}

Similarly, It is tested for the remaining Methods as well.

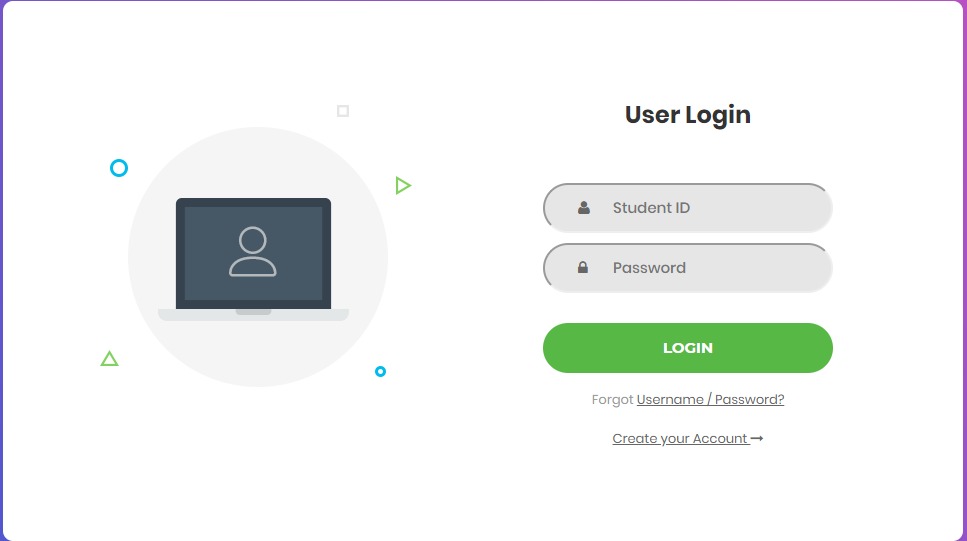
**4. CONTRIBUTIONS**

**4.1 Requirements**

|  |  |
| --- | --- |
| **Contributions** | **Developer Name** |
| **25%** | Sreekanth Vobilishetty |
| **25%** | Akhila Yarlagadda |
| **25%** | Piyusha Varshini Tirukovalluru |
| **25%** | Meghana Reddy Akkati |

**5. USER MANUAL**

By clicking on the link of the url it directs you to the login page, where it has the ID and password options as displayed in this page. It also as the login button which directs you to the main page.



If the student is a new user then he/she should click the create account button which directs you to the new page where you need to enter your details and if the entered id matches with the id in the database it allows you to create the account.

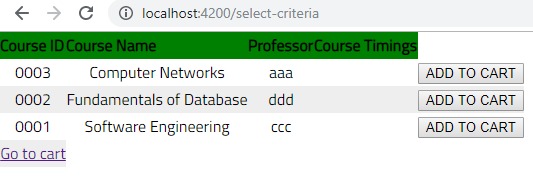
If the user forgets the username/password he/she needs to click the forgot password button which directs you to the new page where you can reset the password by entering the details as shown in this image.

After clicking the login button it directs the main page which looks like the picture shown below, this has the search, enroll, my classes, finances, contact, logout.

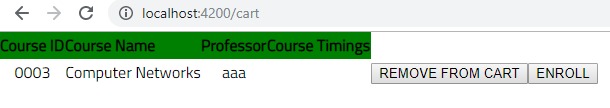
By clicking on the enroll button it dropdown the list which has add, drop, swap, and course catalogue.

The add button here directs you to page to select the degree and the department, and a submit button to submit after choosing the selected options.

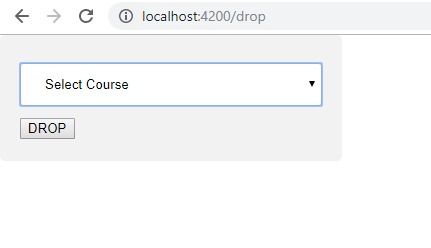
The submit button opens a page with a list of courses, each course name is followed by a add to cart button next to it. After adding the desired courses you can click the o to cart button to view the courses in the cart.



Go to cart button opens a page were you can confirm the enrollment, here you have remove from cart button which removes the course from the cart and enroll button to confirm the enrollment .



Drop button displays a page which has the select course button it dropdown the list of the enrolled courses, now you can click on the course which you wanted drop and select the drop button.



Swap button directs a page which has two dropdown buttons select the course and select the course you want to swap with. After selecting the courses click the swap button to proceed.

Course catalogue will direct you to the page which shows all the courses and description of each course.

**6. INSTALLATION INSTRUCTIONS**

Open spring tool suite on your device, then select course registration and right click on it . Now it dropdown a list in which select the run as. Next select the run the application as spring Boot App.

**7. PEER REVIEW SESSION FEEDBACK**

We have received two feedback points from our peer team

7.1 Not to include the bullet points

In the previous deliverable we have put the bullets points to mention the points. In peer review we got to know that according to IEEE format we shouldn’t use the bullet points.

7.2 Some of the abbreviations are not mentioned

In the previous report we forgot to mention few of the abbreviations, which we got to know durin the peer review session.